Implementasi Struktur Data Stack untuk Pengelolaan Riwayat Penelusuran dalam Bentuk Ekstensi Web Browser Chrome

Muhammad Rizki Andrian Fitra[1] Ega Pratama[2] Muhammad Zidane Al-Kautsar[3] Fatimah Asro Harahap[4] Fanny Ramadhani[5]
Program Studi Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Negeri Medan, Kota Medan, Sumatera Utara, Indonesia[1,2,3,4,5]
Email: andrian25544.4232550001@mhs.unimed.ac.id[1]
pratama.4233550014@mhs.unimed.ac.id[2] zidanekece07.4231250024@mhs.unimed.ac.id[3]
fati.4231250008@mhs.unimed.ac.id[4]

Abstract
Browsing history is a fundamental feature in modern browsers, yet its management remains inefficient in many implementations. This research develops a Chrome extension utilizing the Stack data structure, leveraging its Last In, First Out (LIFO) principle to store, search, delete, and open browsing history URLs efficiently. The system is designed with Chrome's API to monitor tab activities, store data locally, and provide an interactive user interface. The findings demonstrate that the stack-based approach is more efficient than conventional methods. The system also offers users straightforward features for managing history, including search and individual deletion capabilities.
Keywords: Stack Data Structure, Chrome Extension, Browsing History, Chrome API

INTRODUCTION

In the digital era, browsing history is a key feature that facilitates user navigation by enabling quick access to previously visited web pages. However, most browsers employ linear data structures such as lists to manage history, which may lead to inefficiencies in operations like searching or deletion. The Stack data structure presents a promising solution for optimizing such tasks. By adhering to the LIFO principle, the most recent entries are prioritized for retrieval and removal, simplifying the management of temporary data like browsing history. Previous studies have explored the use of linear lists for browsing history storage. Yet, these methods often struggle with efficiency in dynamic data manipulation. Therefore, this study introduces a novel approach using the stack structure to manage browsing history within a Chrome extension, providing enhanced usability and functionality.

Research Objectives:

1. To develop a Chrome extension utilizing the stack data structure for managing browsing history.
2. To integrate features for searching and deleting history entries.
3. To assess the efficiency of the stack-based approach compared to conventional methods.

RESEARCH METHODS

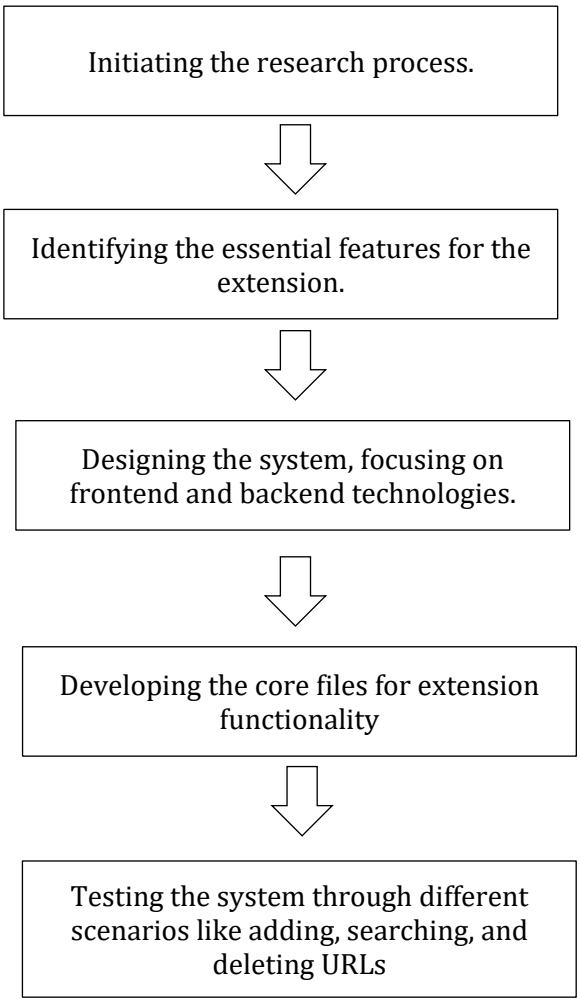
This research adopts a software engineering approach, consisting of the following steps:

1. Requirements Analysis: Features such as stack-based URL storage, search capabilities, individual deletion, and direct navigation were identified as essential.
2. Technology Stack:
   2.1. Frontend: HTML and CSS for user interface design.
   2.2. Backend: JavaScript to manage system logic and integrate Chrome's API.

2.3. Data storage via chrome.storage.local, ensuring persistence and security.

3. System Implementation:
   3.1. Developed the manifest.json file for extension configuration.
   3.2. Created background.js for monitoring tab activities and storing URLs.
   3.3. Built popup.js to handle user interactions through the extension interface.
4. Testing and Validation: Scenarios included adding, searching, deleting, and navigating URLs.

FLOWCHART OF RESEARCH METHOD



SOURCE CODE
Manifest.Json

```
{
 "manifest_version": 3,
 "name": "Browser History Stack Manager",
 "version": "1.2",
 "description": "Automatically tracks browsing history using a stack structure. Includes search and navigation.",
 "permissions": ["tabs", "history", "storage"],
 "host_permissions": ["<all_urls>"],
 "action": {
```

```
"default_popup": "popup.html",
"default_icon": {
"16": "icon16.png",
"48": "icon48.png",
"128": "icon128.png"
},
"default_title": "Manage History Stack"
},
"background": {
"service_worker": "background.js"
},
"icons": {
"16": "icon16.png",
"48": "icon48.png",
"128": "icon128.png"
}
}
```

Popup.html

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>History Stack</title>
 <link rel="stylesheet" href="style.css">
</head>
<body>
 <div class="container">
 <h1>History Stack</h1>
 <input type="text" id="search-input" placeholder="Search URL" />
 <button id="search-btn">Search</button>
 <div id="history-container"></div>
 <button id="clear-stack">Clear Stack</button>
 </div>
 <script src="popup.js"></script>
</body>
</html>
```

Background.js

```
chrome.tabs.onUpdated.addListener((tabId, changeInfo, tab) => {
 if (changeInfo.url) {
 chrome.storage.local.get(["historyStack"], (result) => {
 let historyStack = result.historyStack || [];
 historyStack.push(changeInfo.url); // Tambahkan URL baru ke stack
 chrome.storage.local.set({ historyStack });
```

```
});
}
});
```

Popup.js

```
let historyStack = [];

// Load stack from storage
chrome.storage.local.get(["historyStack"], (result) => {
 historyStack = result.historyStack || [];
 renderStack();
});

// Render the stack in the popup
function renderStack(searchQuery = "") {
 const container = document.getElementById("history-container");
 container.innerHTML = ""; // Clear container
 let filteredStack = historyStack;

 // Filter stack based on search query
 if (searchQuery) {
 filteredStack = historyStack.filter((url) =>
 url.toLowerCase().includes(searchQuery.toLowerCase())
 );
 }

 if (filteredStack.length === 0) {
 container.innerHTML = "<p>No history available</p>";
 return;
 }

 filteredStack.slice().reverse().forEach((url, index) => {
 const div = document.createElement("div");
 div.className = "history-item";
 div.innerHTML = `
 <span class="url" data-url="${url}">${url}</span>
 <button class="delete-btn" data-url="${url}">Delete</button>
 `;
 container.appendChild(div);
 });

 // Add event listeners for delete buttons
 document.querySelectorAll(".delete-btn").forEach((button) => {
 button.addEventListener("click", (event) => {
 const urlToDelete = event.target.getAttribute("data-url");
 historyStack = historyStack.filter((url) => url !== urlToDelete);
 saveStack();
```

```
renderStack();
});
});

// Add event listeners for URL clicks
document.querySelectorAll(".url").forEach((span) => {
span.addEventListener("click", (event) => {
const urlToNavigate = event.target.getAttribute("data-url");
chrome.tabs.create({ url: urlToNavigate }); // Open URL in a new tab
});
});
}

// Save the stack to Chrome storage
function saveStack() {
chrome.storage.local.set({ historyStack });
}

// Clear the entire stack
document.getElementById("clear-stack").addEventListener("click", () => {
historyStack = [];
saveStack();
renderStack();
});

// Search functionality
document.getElementById("search-btn").addEventListener("click", () => {
const searchInput = document.getElementById("search-input").value;
renderStack(searchInput);
});
```

Style.css

```
body {
font-family: Arial, sans-serif;
margin: 0;
padding: 10px;
background-color: #f4f4f9;
color: #333;
width: 200px;
height: 300px;
overflow-y: auto;
}

.container {
max-width: 180px;
margin: auto;
```

```
}

h1 {
font-size: 16px;
text-align: center;
color: #4a90e2;
}

#history-container {
margin: 10px 0;
background: #fff;
padding: 10px;
border: 1px solid #ddd;
border-radius: 4px;
max-height: 150px;
overflow-y: auto;
}

.history-item {
font-size: 12px;
display: flex;
justify-content: space-between;
padding: 5px 0;
border-bottom: 1px solid #ddd;
}

.history-item:last-child {
border-bottom: none;
}

.url {
cursor: pointer;
text-decoration: underline;
color: #4a90e2;
}

.url:hover {
color: #357abd;
}

button {
background-color: #4a90e2;
color: #fff;
border: none;
border-radius: 3px;
padding: 5px;
cursor: pointer;
text-align: center;
```

```
}

button:hover {
background-color: #357abd;
}

#search-input {
width: 70%;
padding: 5px;
margin-right: 5px;
}

#clear-stack, #search-btn {
width: 100%;
margin-top: 5px;
}
```

RESEARCH RESULTS AND DISCUSSION

The Chrome extension successfully implements all the planned features. The results from the research are as follows:

History Storage

URLs are stored in a stack, with the most recent entries accessible at the top. This simplifies retrieval and deletion operations.

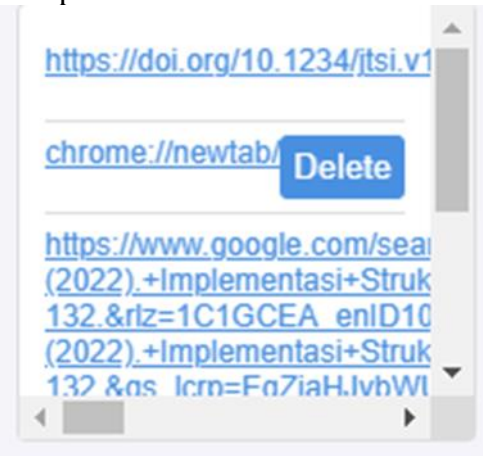


Figure 1. URLs are stored in a stack

Search Functionality

The search feature accurately retrieves URLs matching the user's queries.

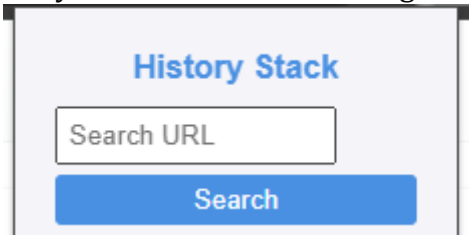


Figure 2. Search Functionality

Individual Deletion

Users can delete specific entries from the stack without affecting other data.

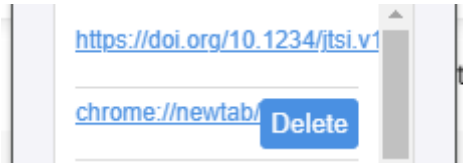


Figure 3. Individual Deletion

Navigation

Users can open URLs directly from the extension, streamlining workflow.

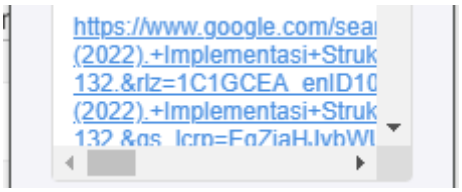


Figure 4. Navigation

.

Discussion

The implementation of the Stack data structure for browsing history management in the Chrome extension has yielded promising results, as outlined in the research findings. This section discusses the processing and significance of the results, the relationship between the research outcomes and the core concepts, and how they compare with previous studies.

1. Processing of Results: The results presented demonstrate that the stack-based system is highly efficient for managing browsing history. The stack structure allows for the most recent browsing URLs to be easily accessed, deleted, and navigated. The search functionality, integrated into the extension, accurately filters URLs based on user input, which enhances the user experience by enabling quick access to specific entries. The individual deletion feature offers flexibility, allowing users to remove particular URLs without affecting the entire history.The storage of URLs in a stack, as seen in the extension's behavior, provides a clear and efficient way to handle the dynamic nature of browsing history. Since the most recent URL is always placed on top, it ensures that the user can retrieve the last visited website quickly, supporting the Last In, First Out (LIFO) principle of stacks.
2. Relationship to Basic Concepts and Hypotheses: The research aligns well with the fundamental principles of stack data structures, particularly the LIFO (Last In, First Out) principle, which is the foundation of the extension's functionality. By applying this principle, the stack enables efficient browsing history management, where the most recently visited URL is prioritized for retrieval and deletion. This is in contrast to traditional list-based approaches, where retrieval and deletion of URLs may be slower and less efficient.The hypothesis that a stack-based approach could optimize browsing history management is confirmed by the results of the study. The stack not only improves efficiency in operations such as search and deletion but also simplifies the task of accessing and navigating through recent URLs, thus enhancing the overall user experience.
3. Comparison with Previous Studies: Several studies have explored the use of linear data structures, such as arrays and lists, for managing browsing history. However, these structures often struggle with inefficiencies when handling tasks such as searching, deleting, or updating entries in large datasets. For example, previous implementations relying on lists

typically require linear scans to search for URLs or delete specific entries, which may not be optimal for frequent browsing sessions.The results of this study contrast with those previous approaches by showing that the stack structure provides faster operations for adding, searching, and deleting URLs due to its inherent LIFO design. Additionally, the extension's integration with the Chrome API and local storage ensures data persistence across sessions, making the stack-based solution both practical and efficient for managing browsing history in a real-world scenario.

4. Significance of Findings: The findings highlight the potential for using the Stack data structure not only in managing browsing history but also in various other areas where data needs to be processed in a LIFO manner. The success of this research demonstrates that the stack can be an efficient and user-friendly solution for handling dynamic data in modern applications. Furthermore, the integration with the Chrome API ensures seamless functionality with the browser, making it a highly effective tool for real-time browsing history management.The system's ability to offer features such as direct URL navigation and history stack clearance also adds significant value to the user experience, making it easy for users to manage their browsing history actively.
5. Future Research Directions: While this research demonstrates the efficacy of using a stack for browsing history management, there are several avenues for future work. One area of exploration could involve integrating advanced analytics to track and visualize browsing patterns over time. This could provide users with insights into their browsing habits and allow for personalized recommendations based on their history.Furthermore, user feedback could be collected to assess the usability and effectiveness of the extension in different real-world scenarios. This could help identify potential areas for improvement, such as the addition of more sophisticated filtering options or enhanced navigation features. Exploring alternative storage mechanisms, such as cloud-based solutions, could also improve the persistence and scalability of the system.

CONCLUSION

In conclusion, the stack-based approach implemented in the Chrome extension successfully addresses the challenges of managing browsing history efficiently. The results show that this solution offers significant advantages over traditional methods, providing users with enhanced functionality and a streamlined browsing experience.This research successfully demonstrates the efficiency of a stack-based approach for browsing history management within a Chrome extension. The extension provides features such as URL storage, search, and deletion, alongside direct navigation. Future research can explore advanced analytics and user insights for further enhancement.

BIBLIOGRAPHY

Fathurahman, R., & Rizqi, Y. (2021). Sistem Informasi Akademik Berbasis Web di SMA Negeri 1 Banjarharjo. *Jurnal Pendidikan Teknik Informatika*, 9(1), 45–50. Retrieved from https://journal.student.uny.ac.id

Pradipta, A. A., Yusman, M., & Shofiana, D. A. (2023). Sistem Manajemen Data Akademik Berbasis Web. *Jurnal Teknologi dan Rekayasa*, 10(3), 65–75. Retrieved from https://jurnal.lldikti4.or.id

Purwanto, H. (2023). Penerapan Sistem Informasi Akademik (SIA) Sebagai Upaya Peningkatan Efektivitas dan Efisiensi Pengelolaan Akademik Sekolah. *Jurnal Teknologi Terapan*, 3(2), 24–31. https://doi.org/10.31884/jtt.v3i2.58

Sofyan, R. (2023). Pengembangan dan Implementasi Sistem Informasi Akademik Menggunakan Framework SISFO Kampus. *Jurnal Teknik Informatika UIN Jakarta*, 15(1), 35–45. Retrieved from https://repository.uinjkt.ac.id

Yassir, H. (2023). Perancangan dan Implementasi Sistem Informasi Akademik Berbasis Scrum. *Jurnal Fokus Elektroda*, 8(3), 171–178. Retrieved from https://elektroda.uho.ac.id